

Automatic Index Optimization combined with Text Clustering by KNN and AHC Algorithm considering Feature Similarities

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the KNN algorithm and the AHC algorithm which consider the feature similarities for automating the index optimization by introducing the text clustering. In the previous work, even if the KNN version was proposed as an approach to the index optimization, it was required to present a domain as a tag of an input text for carrying out the task. In this research, text subgroups are constructed based on their contents through the text clustering, each word in a novice text is classified into one of the three categories by a classifier which corresponds to its most similar cluster. The AHC algorithm and the KNN algorithm which consider the feature similarities are adopted as the respective approaches to the text clustering and the index optimization. The goals of this research are to automate the keyword extraction and to improve the performance of both tasks, using the two proposed algorithms.

I. INTRODUCTION

The index optimization is defined as the process of optimizing the text index for maximizing the information retrieval performance. It is viewed into a classification task where the three categories, expansion, inclusion, and removal, are predefined. The process of index optimization is to index a text into words and classify each word into one of the three categories. Associated words from external sources are added to ones which are classified into expansion, ones which are classified into inclusion are included by themselves, and ones which are classified into removal should be excluded. The index optimization is to reinforce important words through the index expansion and to remove unimportant words through the index filtering.

This research is motivated by the necessity of defining domains for designing the index optimization system. The index optimization is mapped into an independent classification task for each domain. The process of designing the index optimization system is to predefine domains as a list, collect sample examples domain by domain, and allocate a classifier for each domain. The excellent performance was discovered in applying the KNN algorithm and the AHC algorithm which consider the feature similarities respectively to the index optimization and the text clustering. This research is intended to automate the domain decision by connecting the index optimization with the text clustering.

In this research, we propose that the index optimization should relate to the text clustering, and the KNN algorithm and the AHC algorithm which consider the feature similarities should be applied to both tasks. In this research, it is assumed that the text collection is prepared, and each in the collection is associated with words which are classified into one of the three classes. The similarity metric between two numerical vectors which considers the feature similarities is defined, and the KNN algorithm and the AHC algorithm are modified based on the similarity metric. The collection is partitioned into clusters by the text clustering, and the classification task which is mapped from the index optimization is implemented for each cluster. A domain is assigned to an input text by its similarities as the cluster prototypes for selecting a corresponding classifier.

Let us mention some benefits from this research. The poor discrimination among sparse vectors is solved by considering the feature similarities in computing the similarity between two numerical vectors. The performances in the index optimization and the text clustering are improved by adopting the modified versions of the KNN algorithm and the AHC algorithm. The domain predefinition is automated through the text clustering which relates to the index optimization. A domain is automatically decided to an input text; only an input text is presented without its domain for nominating a classifier.

Let us mention some remaining tasks as the further discussions on this research. Only essential features, rather than all features, are involved in computing the feature similarities for reducing the computation complexity. We modify more advanced machine learning algorithms and deep learning algorithms into the versions which consider the feature similarities. We implement the index optimization system which relates to the text clustering as a real program or a library. The index optimization system may be installed in an information retrieval system for improving its performance.

II. PROPOSED APPROACH

A. Similarity Metric

This section is concerned with the process of encoding a word into a numerical vector. Texts are used as features

for encoding a word so, and some are selected from a text collection. A similarity between texts is computed based on their shared words, and it is used as a similarity between features. The similarities among features are considered for computing the semantic similarity between words. In this section, we describe the process of encoding a word into a numerical vector and the process of computing the similarity between words.

The process of encoding words into numerical vectors is illustrated in Figure 1. Texts are gathered as a collection from external sources as feature candidates. Only some texts are selected by a criterion among the gathered texts. In each word, its weights in the texts are computed as elements of the numerical vector. Refer to [3] for studying the process and the selection criteria.

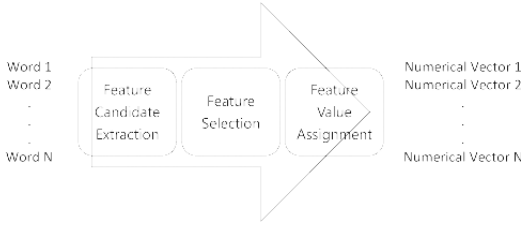


Figure 1. Process of Encoding Words into Numerical Vectors

The frame of computing the similarity between two numerical vectors is illustrated in Figure 2. The two d dimensional vectors and the d features are respectively notated respectively by $\mathbf{x} = [x_1 \ x_2 \ \dots \ x_d]$, $\mathbf{y} = [y_1 \ y_2 \ \dots \ y_d]$, and $f_1, f_2, \dots, f_d$. The feature similarity refers to the similarity between two features, which is notated by $\text{sim}(f_1, f_2)$. The feature value similarity is the similarity between two vectors, $\mathbf{x}$ and $\mathbf{y}$, such as cosine similarity. The similarity between words is computed by considering the two kinds of similarities.

$$\begin{aligned} \mathbf{x} &= [x_1, x_2, \dots, x_d] \\ \mathbf{y} &= [y_1, y_2, \dots, y_d] \end{aligned} \quad \begin{array}{l} \text{Feature} \\ \text{Similarity} \\ \\ \text{Feature} \\ \text{Value} \\ \text{Similarity} \end{array}$$

Figure 2. Frame of Computing Similarity between Numerical Vectors

Let us mention the process of computing the similarity between numerical vectors as the semantic similarity between words. The similarity between features, f_i and f_j , $\text{sim}(f_i, f_j)$, is abbreviated into s_{ij} . The similarity between features is computed by equation (1),

$$s_{ij} = \frac{2 \times |D_i \cap D_j|}{|D_i| + |D_j|} \quad (1)$$

where D_i is the set of words in the feature, f_i , and D_j is the set of words in the feature, f_j , and the similarity matrix

with the d features is constructed as a $d \times d$ square matrix, as shown in equation (2),

$$\mathbf{S} = \begin{pmatrix} s_{11} & s_{12} & \dots & s_{1d} \\ s_{21} & s_{22} & \dots & s_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ s_{d1} & s_{d2} & \dots & s_{dd} \end{pmatrix}. \quad (2)$$

The similarity between two numerical vectors, $\mathbf{x}$ and $\mathbf{y}$, is computed by equation (3),

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^d \sum_{j=1}^d s_{ij} x_i y_j}{d \cdot \|\mathbf{x}\| \cdot \|\mathbf{y}\|}. \quad (3)$$

Equation (3) is used for computing the similarity between a novice word and a sample word in the proposed version of the KNN algorithm.

Let us make some remarks on the encoding process the process of encoding words into numerical vectors and computing the similarity between them. A word is encoded into a numerical vector by the feature candidate extraction, the feature selection, and the feature value assignment. The similarity between features which are given as texts is computed by equation (1). The similarity between numerical vectors which represent words is computed, considering the feature similarities by equation (3). In future, we will consider computing the similarities among some features rather than all features, for improving the computation speed.

B. Feature Similarity based KNN Algorithm

This section is concerned with the version of KNN algorithm which considers the feature similarities. The version was initially proposed as the approach to the text classification by Jo in 2019 ?? . The similarity metric between numerical vectors which was described in the previous section is used for computing the similarity between a novice vector and a training example. The labels of its nearest neighbors are voted with their identical weights for decoding the label of a novice vector. This section is intended to describe the initial version of the KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 3. The labeled words which are gathered as samples are encoded into numerical vectors, and they are notated by a set $Tr = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$. A novice word is encoded into a numerical vector notated by $\mathbf{x}$. Its similarities with the numerical vectors which represent the sample words are computed by equation (3), as $\text{sim}(\mathbf{x}, \mathbf{x}_1), \text{sim}(\mathbf{x}, \mathbf{x}_2), \dots, \text{sim}(\mathbf{x}, \mathbf{x}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The

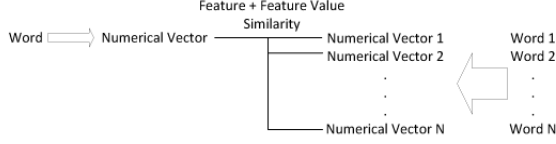


Figure 3. Similarities of Novice Input with Training Examples

training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (4),

$$Nr = \text{Select}_k(Tr, \mathbf{x}), Nr \subseteq Tr \quad (4)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (5),

$$Nr = \{\mathbf{x}_1^{near}, \mathbf{x}_2^{near}, \dots, \mathbf{x}_k^{near}\} \quad (5)$$

The elements in the set, Nr , are used for voting their labels in the next step.

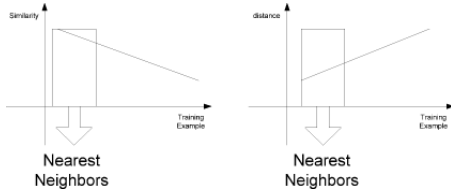


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(\mathbf{x}_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$. The label of the novice item, $\mathbf{x}$, is decided by equation (6),

$$\text{label}(\mathbf{x}) = \underset{j=1}{\overset{m}{\text{argmax}}} \text{Count}(Nr, c_j) \quad (6)$$

The process of deciding the label of a novice item by equation (6), is called voting.

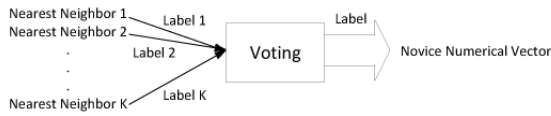


Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the proposed version of KNN algorithm which considers the feature similarities. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest

neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the index optimization system which adopts the KNN algorithm which considers the feature similarities. In Section II-B, we described the proposed version of KNN algorithm as the approach to the index optimization. It is viewed as a classification task which classifies a word into one of the three categories. This section is intended to describe the index optimization system with respect to its function and its architecture.

The process of collecting the same words and encoding them into numerical vectors for implementing the index optimization system is illustrated in Figure 6. The index optimization is interpreted into a domain dependent classification; even a same word is classified differently, depending on the domain. For each domain, an independent classification task is defined. The several domains are decided, and the words which are labeled with one of the three categories exclusively in each domain. It is required to present the domain from a text which is given as the input for doing this task.

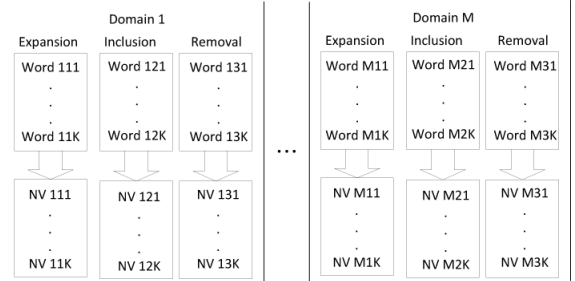


Figure 6. Preparation of Training Examples

The entire system architecture of the index optimization system which is implemented in this research is illustrated in Figure 7. The words which are labeled one of the three categories are collected as the sample words, and they are encoded into numerical vectors by the encoder module. A $d \times d$ similarity matrix which is used for computing the feature similarity is constructed, and the similarities of each word with the sample words which is indexed from a text are computed in the similarity computation module. In the voting module, nearest neighbors are selected, and the category is decided for each word in the text. In the similarity computation module, the ranking module is nested for selecting the nearest neighbors.

The execution process of the proposed system is illustrated as a block diagram in Figure 8. Sample words which are labeled with one of the three categories are collected within a domain, and they are encoded into numerical

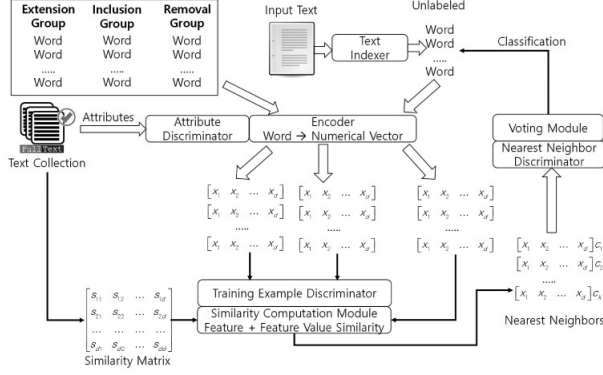


Figure 7. System Architecture

vectors. An input text is indexed into a list of words, and they are also encoded into numerical vectors. The nearest neighbors are selected by the similarity computation, and the label is decided by voting ones of their nearest neighbors for each word. Ones which are classified into expansion require their associated words from external sources, ones which are classified into inclusion are preserved, and ones which are classified into removal are excluded.

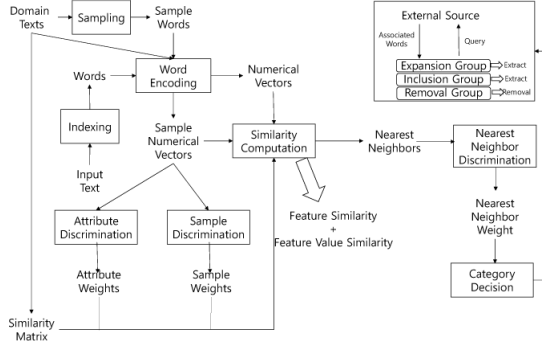


Figure 8. System Flow

Let us some remarks on the index optimization system which is presented in Figure 7 and 8. We need the two collections: the collection of sample words and the collection of texts within each domain. The system is implemented as multiple independent subsystems domain by domain, and it is required to present the domain as a tag for performing the index optimization to a text. The proposed system is described staying in the design step; the source code in Java or Python will be presented in the next research. We need to implement the additional module for expanding the index to the words in the expansion group.

D. Connection with Text Clustering

This section is concerned with the connection of the index optimization with the text clustering. In the previous section, we mentioned the index optimization as an instance

of domain dependent classification. The domains are automatically constructed from the text collection by clustering texts. The AHC algorithm which is proposed in [2] is used for implementing the text clustering. This section is intended to describe the index optimization system which is connected with the text clustering for automating the construction of domains.

The architecture of the text clustering system which was proposed in [2] is illustrated in Figure 9. The texts in the group are encoded into numerical vectors, and the AHC algorithm which is proposed in [2] is adopted as the approach. It starts with singletons as many as items, computes the similarities of all possible cluster pairs, and merge the cluster pair with its highest similarity into one cluster. The two steps are iterated until the number of clusters reach the desired number. We may consider replacing the AHC algorithm by its variants for improving the speed and the performance.

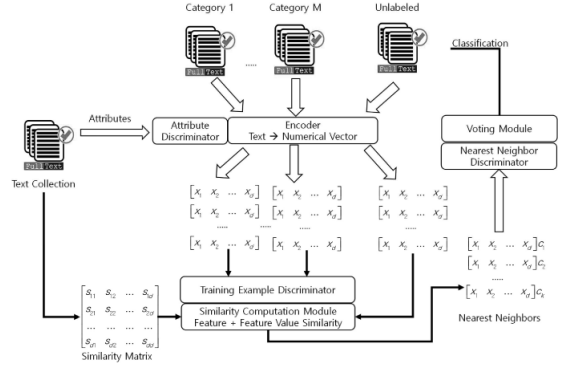


Figure 9. Text Clustering System

The connection of the index optimization with the text clustering is illustrated in Figure 10. The M domains are defined by clustering texts in a group, initially and automatically. A novice text is given as the input, and its domain, domain D , is decided by its similarities with domains. A classifier which corresponds to domain D is nominated and classify each word in a novice text into expansion, inclusion, or removal. The M index optimization systems are viewed as independent ones, each of which classifies each word into one of the three categories.

The execution flow of index optimization which is connected with the text clustering is illustrated in Figure 11. Unlabeled texts are clustered into subgroups, each of which is a domain. Sample words which are labeled with expansion, inclusion, or removal are generated from each domain. These sample words are provided for training the classifier in each index optimization system. Each classifier is used for judging the importance level of each word which is indexed from the input text.

Let us make some remarks on the connection of the index optimization with the text clustering. It is the process

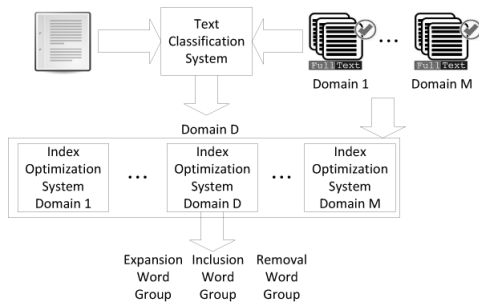


Figure 10. Index Optimization System connected with Text Clustering

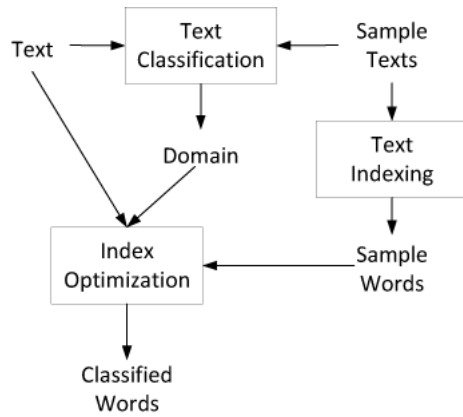


Figure 11. System Flow of Index Optimization connected with Text Clustering

of segmenting a text group into subgroups based on their content-based similarities. The role of text clustering is to define the domains initially in the architecture of connecting the index optimization with the text clustering. In each domain, sample words are generated, and its corresponding classifier is trained with them. In the future research, we consider using the index optimization for clustering texts in the opposite direction.

REFERENCES

- [1] T. Jo, "Index Optimization in News Articles using Feature Similarity based K Nearest Neighbor", 106-109, The Proceedings of 17th Int'l Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government, 2018.
- [2] T. Jo, "Clustering Texts using Feature Similarity based AHC Algorithm", 5993-6003, Journal of Intelligent and Fuzzy Systems, 35, 2018.
- [3] T. Jo, "Text Classification using Feature Similarity based K Nearest Neighbor", 13-21, AS Medical Science, 3, 4, 2019.